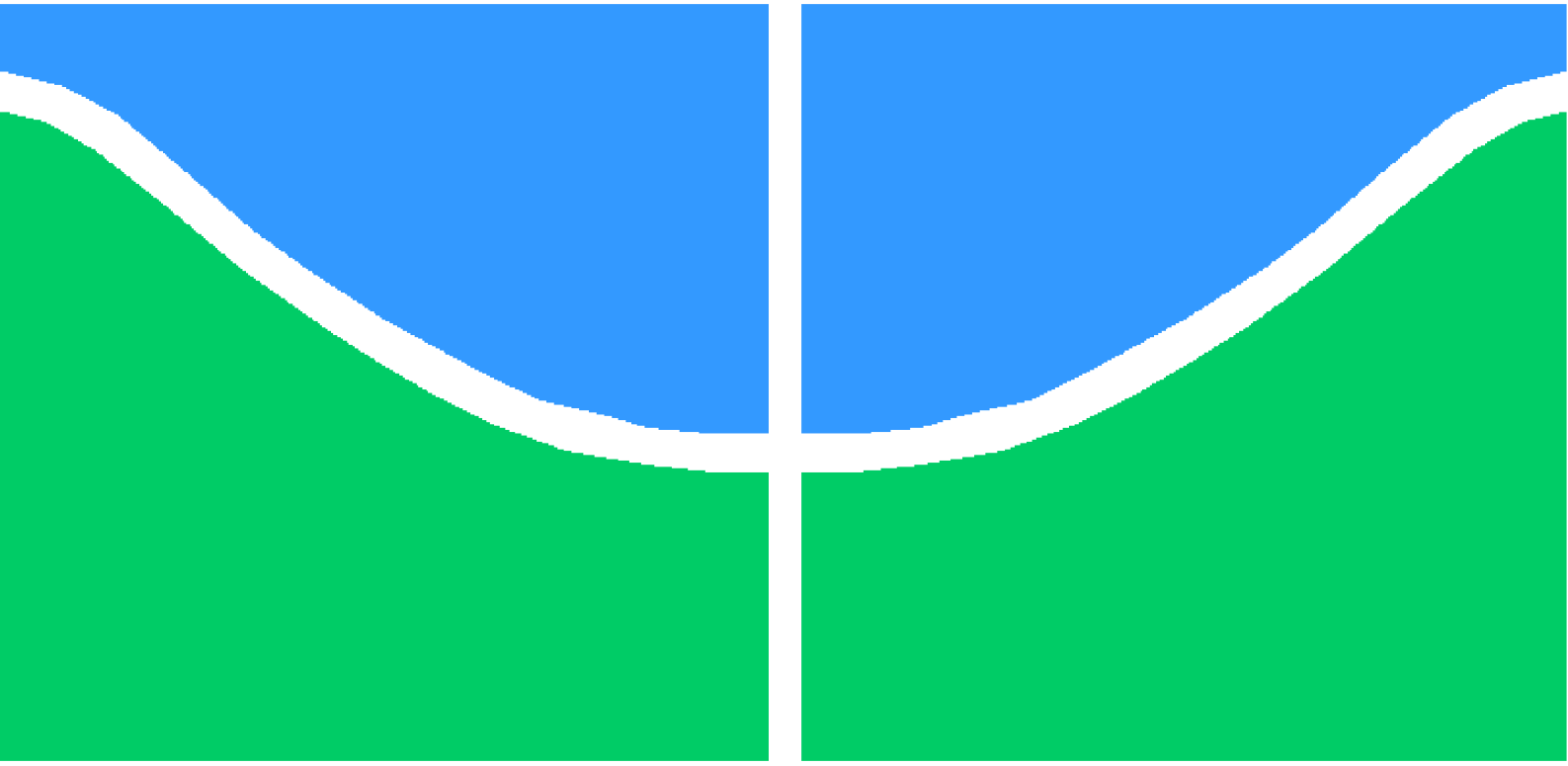


Universidade de Brasília – UnB
Faculdade UnB Gama – FGA
Engenharia de *Software*

Alergio: Sistema de Gerenciamento de Reações Adversas

Autor: Victor Matias Navarro
Orientador: Carla Silva Rocha Aguiar

Brasília, DF
2018

Victor Matias Navarro

Alergio: Sistema de Gerenciamento de Reações Adversas

Monografia submetida ao curso de graduação em Engenharia de *Software* da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de *Software*.

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Orientador: Carla Silva Rocha Aguiar

Brasília, DF

2018

Victor Matias Navarro
Alergio: Sistema de Gerenciamento de Reações Adversas/ Victor Matias Navarro. – Brasília, DF, 2018-
64 p. : il. (algumas color.) ; 30 cm.

Orientador: Carla Silva Rocha Aguiar

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB
Faculdade UnB Gama – FGA , 2018.

1. Reação adversa. 2. Medicamentos. I. Carla Silva Rocha Aguiar. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Alergio: Sistema de Gerenciamento de Reações Adversas

CDU 02:141:005.6

Errata

Victor Matias Navarro

Alergio: Sistema de Gerenciamento de Reações Adversas

Monografia submetida ao curso de graduação em Engenharia de *Software* da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de *Software*.

Trabalho aprovado. Brasília, DF, 28 de março de 2019:

Carla Silva Rocha Aguiar
Orientador

Fábio Macedo Mendes
Convidado 1

Renato Coral Sampaio
Convidado 2

Brasília, DF
2018

Aos companheiros de vida que eu tenho o prazer de chamar de família.

Agradecimentos

Esse trabalho não teria sido possível sem meu pai e irmã, que me auxiliaram através de várias conversas. Minha mãe e meu irmão foram igualmente importantes por aturarem meus surtos durante o semestre, principalmente na última fase.

O Laboratório Avançado de Produção, Pesquisa e Inovação em *Software* (LAPPIS) também foi extremamente importante na minha formação, dando-me a experiência necessária para realizar este trabalho.

Por fim, devo agradecimentos à Carla Rocha, a quem admiro muito e é minha orientadora em todos os aspectos possíveis de uma vida.

“Na vida, não há nada a temer,
mas a entender.“
(Marie Curie)

Resumo

Durante um tratamento médico, é vital que o agente de saúde tenha acesso ao maior número possível de informações sobre o paciente, como por exemplo as alergias e reações adversas apresentadas pelo mesmo. Mais do que apenas acesso, é preciso que as informações estejam completas e atualizadas. Em posse desses dados, o tratamento pode ocorrer de maneira a evitar eventos adversos que prejudiquem a saúde do paciente. Este trabalho propõe uma iniciativa de *e-Health* e *mHealth* na forma de um registro pessoal de saúde que permite a um usuário cadastrar suas reações alérgicas e adversas, de forma que essas informações estejam sempre atualizadas. Além disso, a aplicação permitirá a interação com os sistemas de registro eletrônico de saúde presentes nos hospitais para disponibilizar as informações cadastradas em seu banco de dados, sendo que essa comunicação entre os sistemas atenderá a diversos critérios de segurança definidos pela Sociedade Brasileira de Informática em Saúde.

Para garantir o funcionamento supracitado, uma API é responsável por salvar os dados, que podem ser disponibilizados para outros sistemas, ou mesmo incrementados pelos mesmos. No âmbito de estabelecer uma comunicação com o usuário final da aplicação, um Progressive Web App servirá como interface para os variados tipos de usuários da aplicação, garantindo que um cidadão possa inserir suas alergias pelo seu smartphone ou pelo computador.

Palavras-chaves: alergia, reação adversa, medicamentos, registro pessoal de saúde, registro eletrônico de saúde.

Abstract

During a medical treatment, it is vital that the clinicians have access to as much information as possible about the patient, for instance about the allergies and adverse reactions presented by him. More than just access, those informations need to be complete and up to date. Having such data, the treatment may continue in such a way to prevent adverse events that might impair on the patient's health. This project proposes an e-Health and mHealth initiative in the shape of a personal health record that allows a user to register his allergic and adverse reactions in a way that those informations are always up to date. Besides that, the application will allow interaction with electronic health records used in hospitals to share these informations and this interaction will abide by several security criteria defined by Sociedade Brasileira de Informática em Saúde.

To ensure the proposal above, an API is responsible for saving data that can be viewed by other systems, or even be incremented by them. As for establishing a communication with the user, a Progressive Web App will act as interface for all types of user in the system, making sure that a citizen can insert his allergies through his smartphone or his personal computer.

Key-words: allergy. adverse reaction, medicines, personal health record, electronic health record.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

HTTP	*Hypertext Transfer Protocol*
REST	*Representational State Transfer*
API	*Application programming interface*
JWT	*JSON Web Token*
Anvisa	Agência Nacional de Vigilância Sanitária
CI	*Continuous integration*
CD	*Continuous deployment*

Sumário

1 Introdução

1.1 Contextualização

A Organização Mundial da Saúde define um medicamento como um produto para diferentes fins, entre eles diagnóstico e tratamento de doenças (World Health Organization, 2002). Os medicamentos apresentam, cada um, pelo menos um princípio ativo ou fármaco, que por sua vez é definido pela (ANVISA, 2018) como uma substância que tenha efeitos farmacológicos e é utilizada para o benefício do paciente.

Porém, as pessoas podem apresentar uma reação adversa ao princípio ativo utilizado em um medicamento, de forma que o estado dela pode piorar. Nesses casos, é importante que se tenha conhecimento do princípio ativo que despertou a reação de forma a evitar outros medicamentos que o utilizem.

De acordo com (NAGAO-DIAS et al., 2004), as reações alérgicas apresentam alta morbimortalidade e é fundamental que os profissionais envolvidos no tratamento de algum paciente sejam notificados das reações alérgicas que o paciente apresenta.

Dados de 2004 fornecidos por (CASSIANI et al., 2004), após observação dos sistemas de medicações de quatro hospitais brasileiros, indicam que as informações referentes à alergias encontram-se disponíveis em prontuários e não estão completas ou atualizadas. Adicionalmente, até a data da publicação do artigo, nenhum desses hospitais utilizava sistemas informatizados para cadastro dessas informações.

1.1.1 Justificativa

As reações adversas à medicamentos podem resultar em diversas consequências, desde prolongamento de hospitalização à morte. Nesses casos, considera-se o ocorrido como um evento adverso severo (World Health Organization, 2002). Porém, de acordo com (BERND, 2010), reações alérgicas podem provocar também, por exemplo:

- Irritação cutânea, que causa coceira;
- Febre;
- Erupções cutâneas;
- Anafilaxia;

Por esse motivo, é importante concentrar esforços em iniciativas que previnam essas reações de ocorrerem. De uma forma geral, o compartilhamento de informações

sobre as reações adversas de um indivíduo constitui uma forma razoável de prevenção de casos de reações alérgicas ou adversas.

Esse compartilhamento deve ocorrer nas condições mais adversas, como quando o paciente está desacordado ou se encontra em outra cidade que possivelmente não há o registro do mesmo.

1.2 Objetivos

1.2.1 Objetivo Geral

A principal motivação deste projeto de conclusão de curso é propor, implementar e validar um sistema que permita o gerenciamento das reações adversas de um indivíduo, de forma a torná-las acessíveis pelo mesmo e por profissionais da área de saúde.

Além disso, por ter o indivíduo como um dos principais atores do sistema, pretende-se atingir a confiabilidade e constante atualização dos dados presentes na aplicação.

1.2.2 Objetivos Específicos

- Atender a um conjunto pré-definido de requisitos de segurança definidos para sistemas com informações hospitalares;
- Implementar um sistema de gerenciamento de informações acerca de reações adversas;

2 Referencial Teórico

Neste capítulo, serão abordados tópicos necessários para o entendimento da solução proposta. Esses tópicos possuem duas origens diferentes e são explorados sequencialmente.

O primeiro tópico é a área de saúde, que apresenta conceitos como reação adversa e alérgica, *e-Health* e *m-Health*. O próximo é acerca de *software*, com modelos de comunicação entre programas, REST, HTTP e segurança em comunicação entre processos.

Por fim, serão apresentadas soluções existentes, relacionando suas vantagens e desvantagens.

2.1 Reação Alérgica e Reação Adversa

Os medicamentos são uma parte importante na medicina. Eles possibilitam tratar doenças que antes eram fatais ou de difícil tratamento (MELO; RIBEIRO; STORPIRTIS1, 2006). Porém, ministrar um medicamento equivocadamente pode ocasionar sérios problemas, como reações alérgicas. Entretanto, é necessário distinguir reações alérgicas de reações adversas.

Definicão 1. *A OMS classifica reações alérgicas como reação do sistema imunológico de uma pessoa à uma substância (World Health Organization, 2002).*

Definicão 2. *Reação adversa, por sua vez, é qualquer efeito inesperado e não terapêutico de uma substância em algum indivíduo. As reações alérgicas são, portanto, um subgrupo das reações adversas (World Health Organization, 2002).*

Um evento adverso é caracterizado, então, por qualquer ocorrência que não é benéfica ao paciente durante o tratamento, sem necessariamente ser causado pelo mesmo. Esses eventos podem ser graves, ocasionando até morte (World Health Organization, 2002).

Em um tratamento, é necessário que o médico tenha conhecimento prévio do histórico de reações adversas de um paciente, evitando o uso de medicamentos que possam provocar algum dano. Um sistema hospitalar, definido na próxima seção, pode facilitar esse processo de informar o médico acerca do histórico do paciente.

2.2 Sistemas na Saúde

De acordo com (Sociedade Brasileira de Informática em Saúde, 2013), os sistemas eletrônicos de informação tem sido utilizados para substituir registros em papéis em di-

versas áreas e a saúde não é uma exceção à essa regra. Tecnologias vem sendo utilizadas para substituir os prontuários em papéis nos hospitais.

Com isso, entre os tipos de sistemas utilizados na saúde, dois devem ser explicados: Sistema de Registro Eletrônico em Saúde (S-RES) e Registro Pessoal de Saúde. Os dois são definidos na tabela 1 (Sociedade Brasileira de Informática em Saúde, 2013; VICENTINI et al., 2010).

Sistema	**Decrição**
Sistema de Registro Eletrônico de Saúde	Sistema que permite a inserção e manutenção de dados médicos relacionados à um paciente.
Registro Pessoal de Saúde	Sistema que contém dados médicos relacionados à um paciente, porém com o enfoque no paciente, visto que o mesmo insere e gerencia as informações presentes no sistema.

Tabela 1: Definições de S-RES e Registro Pessoal de Saúde (Sociedade Brasileira de Informática em Saúde, 2013; VICENTINI et al., 2010).

Os conceitos de *e-Health* e *mHealth* estão relacionados ao uso de sistemas na área de saúde.

2.3 e-Health e mHealth

e-Health e *mHealth* são conceitos relacionados à saúde que possuem diversas definições (World Health Organization, 2011; ALVAREZ, 2002). Porém, um entendimento comum sobre *e-Health*, de acordo com (ALVAREZ, 2002), é o uso de tecnologias de informação para fornecer funcionalidades de saúde.

Já (OH et al., 2005) apresenta 51 definições diferentes para *e-Health*. Há a referência para saúde e tecnologia em todas as definições, sendo que *internet* é mencionada na maioria delas. "A integração da *internet* na prática médica"(WATSON, 2004), é a definição utilizada no contexto do projeto.

Por sua vez, *mHealth* é uma parte específica dentro de *e-Health* que preocupa-se com a melhora da prática médica através da utilização de dispositivos móveis e suas respectivas implicações tecnológicas (World Health Organization, 2011).

2.4 Visão Geral

O sistema proposto para resolver a problemática apresentada neste documento, disponibilizando informações alérgicas de uma pessoa, enquadra-se como um Registro Pessoal de Saúde, visto que o usuário será responsável por inserir as informações rela-

cionadas às reações adversas e alérgicas apresentadas pelo mesmo. Com esse modelo, espera-se que os dados estejam sendo constantemente atualizados e completos.

Ao disponibilizar essas informações, objetiva-se criar uma integração entre o registro pessoal de saúde aqui proposto com os mais diversos sistemas de registro eletrônico de saúde utilizados nos centros de saúde. O usuário é dono de suas informações, responsável pela corretude e completude das mesmas, e essas informações são divididas com os médicos, enfermeiros e outros agentes que as necessitam, tanto a nível do aplicativo proposto quanto a nível de qualquer sistema que deseja integrar-se com este projeto. Basta apenas ter a devida autorização para recuperar as informações na base de dados.

O modelo de comunicação utilizado para estabelecer essa disponibilidade de informações para os agentes de saúde pode ser visualizado na Figura 1.

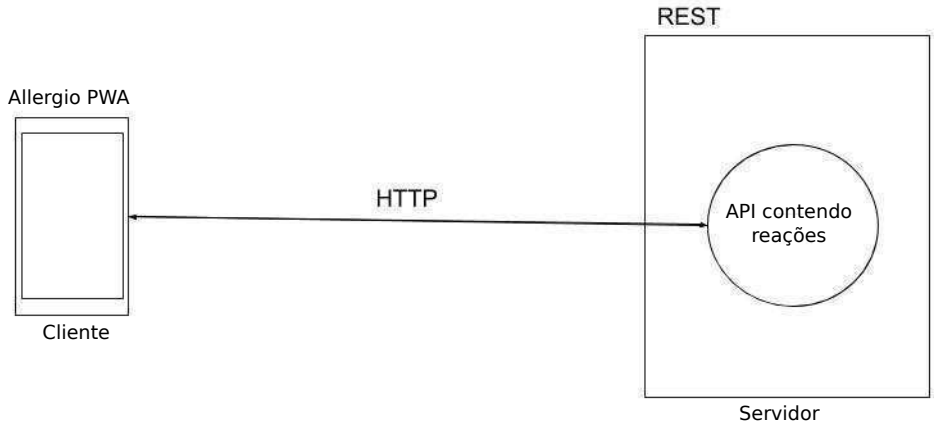


Figura 1 – Interação entre o PWA e API

É importante ressaltar que essa comunicação deve ocorrer de acordo com os princípios apresentados na tabela 5.

Os conceitos mostrados na Figura 1 serão explicados em detalhes nas próximas seções.

2.5 Modelo Cliente-Servidor

Um aspecto fundamental na interação dos componentes mostrada na Figura 1 é a comunicação entre os mesmos. Por utilizarem *internet* para comunicarem-se, os componentes formam uma aplicação de rede.

Segundo (KUROSE; ROSS, 2010), uma aplicação de rede é composta por processos que se comunicam através do uso da rede para enviar mensagens uns para os outros. Nesse

contexto, há duas arquiteturas de aplicações de redes muito utilizadas: P2P e cliente-servidor.

A arquitetura de cliente-servidor consiste em dois processos: um que inicia a comunicação, chamado cliente e outro que espera uma mensagem para enviar uma resposta, chamado servidor (KUROSE; ROSS, 2010).

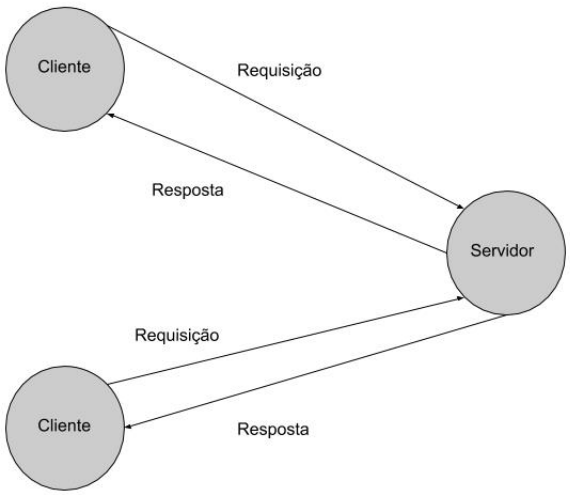


Figura 2 – Comunicação utilizando cliente-servidor.

Neste projeto, essa arquitetura será utilizada, sendo que o processo cliente consiste de um aplicativo que consumirá informações de uma *Application Programming Interface* (API), que é o processo servidor. O modelo cliente-servidor é um dos requisitos do estilo arquitetural *Representational State Transfer* (REST), introduzido na próxima seção.

2.6 *Representational State Transfer*

Um estilo arquitetural é composto por uma série de restrições sobre os elementos de um sistema de forma que fique definido a responsabilidade de cada um deles e como deve ser a interação entre os mesmos (FIELDING, 2000).

REST é um estilo arquitetural criado visando estabelecer procedimentos, responsabilidades e interações para sistemas distribuídos com foco em transferência de arquivos de hipermídia. Possui, basicamente, seis restrições descritas na tabela 2(FIELDING, 2000).

Restrição	Descrição
Cliente-Servidor	Cliente e servidor são entidades independentes que podem ser desenvolvidas em paralelo, visto que o formato das mensagens entre eles são padronizados.
Sem Estado	O servidor não deve manter nenhuma informação sobre o cliente. Cada requisição deve conter informação o suficiente para ser corretamente interpretada e respondida pelo servidor
Cache	Deve haver a possibilidade de guardar as respostas das requisições em uma memória cache
Uniform Interface	A interface entre o cliente e o servidor deve ser bem definida, de forma a não haver um grande acoplamento entre eles
Layered System	Os sistemas podem ser compostos em camadas diferentes, sendo que uma camada só conhece outra a qual interage diretamente
Code on Demand	Única restrição opcional do estilo, permite o download de funcionalidades do servidor para o cliente

Tabela 2: Restrições do REST.

O estilo também prega o desenvolvimento orientado a recurso. Qualquer informação que pode ser nomeada pode ser considerada um recurso. Para diferenciar um recurso de outro, há o que se chama de *Unified Resource Identificator* (URI) (FIELDING, 2000).

Especifica, também, que a troca de informações entre o cliente e o servidor é feita a partir de representações dos recursos. Uma representação é a forma com a qual um recurso será visualizado (FIELDING, 2000). Um exemplo de representação é o *JavaScript Object Notation* (JSON).

Um sistema desenvolvido que atenda à todas as restrições mencionadas pode ser chamado de RESTful. É importante salientar que não é objetivo do trabalho produzir uma API RESTful, isto é, que cumpra exatamente todos os requisitos definidos pelo estilo. Em vez disso, utiliza-se o estilo como fonte de boas práticas que podem ser utilizadas de acordo com a necessidade. O estilo necessita de um protocolo de camada de aplicação, sendo que o utilizado para este trabalho de conclusão de curso é o *Hypertext Transfer Protocol* (HTTP).

2.7 *Hypertext Transfer Protocol*

HTTP é um protocolo da camada de aplicação para transferência de informações de hipermídia, utilizando o modelo cliente-servidor como base para a comunicação entre os processos(KUROSE; ROSS, 2010).

Essa comunicação é estabelecida por mensagens definidas pelo protocolo. O cliente envia uma mensagem de requisição e recebe uma mensagem de resposta do servidor, sendo que o último não mantém dados sobre quem enviou a requisição, o que torna o HTTP um protocolo sem estado ou *stateless* (KUROSE; ROSS, 2010).

As mensagens de requisição e de resposta possuem, respectivamente, um campo para o método da requisição e o código de estado da resposta (KUROSE; ROSS, 2010). Alguns métodos mais comuns, especificados por (BERNERS-LEE et al., 1999), encontram-se na tabela 3.

Método	Propósito
GET	Recuperar uma informação do servidor
HEAD	Recuperar metadados sobre uma informação
POST	Cadastrar uma informação no servidor
PUT	Atualizar uma informação
PATCH	Atualizar parcialmente uma informação
DELETE	Deletar uma informação

Tabela 3: Métodos do HTTP.

Nas respostas, (BERNERS-LEE et al., 1999) define diversas classes de estado. Cada classe contém vários códigos de estado espécificos, sendo que a classe apenas define, de forma geral, o resultado de uma requisição. Mais informações sobre o mesmo podem ser obtidas pelo código específico. As classes encontram-se definidas na tabela 4.

Classe	Propósito
1xx	Atuar como uma resposta intermediária, indicando a necessidade de continuar a requisição. Pode haver a necessidade de pequenas alterações
2xx	Indicar que a requisição foi bem-sucedida
3xx	Indicar que alterações são necessárias para que a requisição seja bem-sucedida
4xx	Indicar algum erro na parte do cliente. Entra nessa classe, por exemplo, códigos que indicam que um recurso não foi encontrado ou que o cliente não tem acesso àquele recurso
5xx	Indicar um erro na parte do servidor, informando que o mesmo não é capaz de processar a requisição

Tabela 4: Classes de estado.

O HTTP pode ser utilizado sobre uma camada adicional de segurança, chamada *Secure Sockets Layer* (SSL). Há uma nova versão com discretas modificações dessa camada, denominada *Transport Layer Security* (TLS). O HTTP sobre SSL/TSL, também conhecido por HTTPS, tem o propósito de fornecer proteção aos dados sendo trocados entre o cliente e o servidor, com base nos princípios definidos na tabela 5 (KUROSE; ROSS, 2010).

Princípio	Descrição
Confidencialidade	Ninguém, além das duas entidades, pode ter conhecimento sobre o conteúdo das mensagens trocadas entre os mesmos
Integridade	Deve ser possível garantir que a mensagem não foi alterada por nenhuma fonte externa à comunicação
Autenticação	Deve ser possível identificar que as entidades são quem elas clamam ser

Tabela 5: Princípios de uma comunicação segura.

2.8 Trabalhos Relacionados

Nesta seção, são introduzidos sistemas que possuem temáticas similares às do sistema proposto. Essas similaridades ocorrem tanto à nível dos sistemas tratarem de reações alérgicas quanto à nível de serem sistemas que possuem como ator central o cidadão comum.

2.8.1 Alergia a Medicamentos

Aplicativo que permite que um usuário cadastre remédios que deseja evitar. Além disso, permite a busca por outro medicamento de forma a indicar se o consumo dele é seguro ou não para o usuário.

Permite, também, adicionar o nome do médico que trata o usuário, bem como seu telefone para o caso de eventuais emergências relacionadas ao uso indevido de um medicamento.

As desvantagens desse aplicativo residem no fato de não compartilhar informações com outros sistemas. Além disso, apesar de permitir o cadastro de um médico que pode receber alertas em caso de uma emergência, os agentes de saúde não têm como saber as alergias de um usuário.

O aplicativo pode ser acessado no seguinte *link*: <https://play.google.com/store/apps/details?id=br.com.imabrasil.alergiaamedicamentos&hl=pt_BR>.

2.8.2 Microsoft Healthvault

Registro pessoal de saúde *web* para armazenar informações relacionadas à saúde de um indivíduo e fornecer diversos serviços para melhorar a qualidade de vida do mesmo.

Uma possibilidade que o sistema oferece é de criar um perfil de emergência que disponibiliza informações como medicamentos que o usuário está utilizando, alergias, data de nascimento e tipo sanguíneo.

Além disso, disponibiliza essas informações para que outros sistemas possam utilizar os dados presentes na aplicação.

As desvantagens do Healthvault, relativas ao contexto, está na impossibilidade de verificar se o consumo de um medicamento é seguro. Além disso, uma diferença em relação ao sistema proposto está no fato de que o sistema proposto utiliza a lista de medicamentos criadas pela Anvisa para orientar o cadastro de alergias de um usuário.

Outras informações sobre o Healthvault podem ser encontradas no seguinte *link*: <https://international.healthvault.com/br/pt>.

2.8.3 Apple Healthkit

Healthkit é uma solução de registro pessoal de saúde da Apple que permite que aplicativos de terceiros possam ler ou escrever, com a devida autorização do usuário, dados no aplicativo "Saúde"presente no *smartphone*.

Há no aplicativo Saúde a possibilidade também de criar uma ficha médica que pode ser visualizada no *smartphone* sem o desbloqueio do mesmo. No caso de uma emergência, pessoas próximas podem utilizá-la para saber como proceder melhor.

As desvantagens do Healthkit são parecidas com as do Healthvault, visto que são *softwares* com propósito mais amplo do que apenas alergia, resultando na perda de algumas funcionalidades em relação à reações adversas importantes, como verificar o consumo de um medicamento. Além disso, também não utiliza a base de dados medicamentosa regulada pela Anvisa.

Outras informações sobre o Apple Healthkit podem ser encontradas no seguinte *link*: <https://developer.apple.com/healthkit/>.

2.8.4 Consulta Remédios - Economize na Farmácia com o CR

Aplicativo que permite que um usuário pesquise, através do código de barras ou do nome de um medicamento, descobrir informações relevantes como:

- Princípio ativo
- Bula
- Classe terapêutica do medicamento

Esse aplicativo tem como principal foco apenas informar o usuário acerca dos aspectos de um medicamento, não permitindo nem o cadastro de uma alergia. Entretanto, é importante ser citado pelo seu bom trabalho em disponibilizar de forma acessível os dados disponibilizados pela Anvisa acerca dos remédios.

O aplicativo pode ser acessado pelo seguinte *link*: <https://play.google.com/store/apps/details?id=com.consultaremedios&hl=pt_BR>.

2.8.5 Meu DigiSUS

Meu DigiSUS é uma plataforma apoiada oficialmente pelo Ministério da Saúde que oferece ao cidadão uma integração com diversas funcionalidades, dentre as quais verificar e avaliar consultas, verificar a lista de transplantes e medicamentos pelo Farmácia Popular.

Em uma seção do aplicativo, também é possível cadastrar as reações adversas que um usuário possui, podendo adicionar até mesmo alergias alimentares, funcionalidade que entra no escopo do projeto proposto neste documento.

O aplicativo pode ser acessado pelo seguinte *link*: <https://play.google.com/store/apps/details?id=br.gov.datasus.cnsdigital&hl=pt_BR>.

2.8.6 Comparação das Soluções

Nesta seção, avaliam-se as soluções supracitadas com os critérios de avaliação definidos na tabela 6.

Critério de Avaliação	**Decrição**
CA1	O sistema permite ao usuário cadastrar alergias?
CA2	O sistema permite que o usuário verifique se um medicamento é seguro para consumo?
CA3	O sistema permite que outros sistemas ou usuários acessem, através do uso de um protocolo de comunicação, as informações nele contidas?
CA4	O sistema apresenta informações acerca dos remédios, além de nome e princípio ativo, para o usuário?

Tabela 6: Critérios de avaliação.

A solução proposta, Allergio, é capaz de cadastrar reações adversas de um usuário, bem como cadastrar princípios ativos que ele está consumindo com o intuito de fornecer informações mais adequadas ao tratamento.

Adicionalmente, permite também que o usuário confira se um medicamento pode ser consumido, baseado no cadastro das reações feito anteriormente. Por fim, disponibiliza essas informações para agentes de saúde que necessitem das mesmas, em qualquer lugar.

A tabela 7 apresenta a avaliação de todos os sistemas, incluindo o Allergio, de acordo com os critérios.

	CA1	CA2	CA3	CA4
Alergia a Medicamentos	Sim	Sim	Não	Não
Microsoft Healthvault	Sim	Não	Sim	Não
Apple Healthkit	Sim	Não	Sim	Não
Consulta de Remédios, Bula, Medicamentos, Posologia	Não	Não	Não	Sim
Allergio	Sim	Sim	Sim	Não

Tabela 7: Comparação das soluções.

Além desses critérios, é importante ressaltar que o Allergio contém uma abrangência de informações maior do que a dos outros sistemas apresentados, contendo:

- Informações acerca dos hospitais brasileiros;
- Informações acerca dos agentes de saúde brasileiros;
- Base de dados preparada para registrar os medicamentos de acordo com o padrão de cadastro da Anvisa;

Todas essas características só são possíveis por ser um sistema pensado para uma problemática específica, que não tenta ser abrangente para tantos casos de uso diferentes, como é o caso das outras soluções supracitadas.

3 Metodologia

Este capítulo visa introduzir a solução proposta, bem como seu funcionamento, e está estruturado da seguinte forma:

- Metodologia;
- Requisitos;
- Solução Proposta;
- Desenvolvimento da Solução;

3.1 Metodologia

O método escolhido para guiar o desenvolvimento da solução é o Kanban. Utilizado por times ágeis, tem como características marcantes a transparência e a fácil visualização do trabalho feito por uma equipe de desenvolvimento (RADIGAN, 2018).

A transparência é alcançada através de dois elementos (RADIGAN, 2018):

Cartões: Cada cartão contém uma tarefa para o desenvolvimento do *software*

Quadro do Kanban: Quadro que contém ao menos três colunas para os cartões: A Fazer, Fazendo e Feito. Os cartões serão movidos da esquerda para a direita no quadro, evidenciando o desenvolvimento.

Além disso, outros benefícios podem ser observados através da utilização do método (RADIGAN, 2018):

Flexibilidade: A priorização das tarefas é feita na primeira coluna do quadro, não influenciando o trabalho que está sendo realizado no momento. Isso permite que exista uma priorização que pode ser constantemente alterada sem interromper o fluxo de trabalho.

Desenvolvimento Contínuo: Por não contar com períodos de tempo fixo onde o trabalho é realizado (como no Scrum com as *sprints*), o fluxo de trabalho no Kanban é contínuo.

Entrega Contínua: Por contar com tarefas separadas e priorizadas que serão desenvolvidas pelo fluxo do quadro, ao fim de cada tarefa há uma entrega que agrega valor ao produto final e pode ser vista por todos os interessados no projeto.

No desenvolvimento da solução, utilizará-se o seguinte processo:

1. São definidas as *features* de desenvolvimento e traça-se uma prioridade entre elas. Todas são colocadas de acordo na coluna "À Fazer";
2. A *feature* com maior prioridade é movida para a coluna "Fazendo";
3. Uma *branch* é criada para o desenvolvimento da *feature*;
4. A *feature* é desenvolvida e testada;
5. A *branch* é incorporada na branch *Master*, que conterá todas as *features* desenvolvidas, testadas e estáveis do projeto;
6. O cartão é movido para "Feito";

Esse processo pode ser observado na seguinte figura:

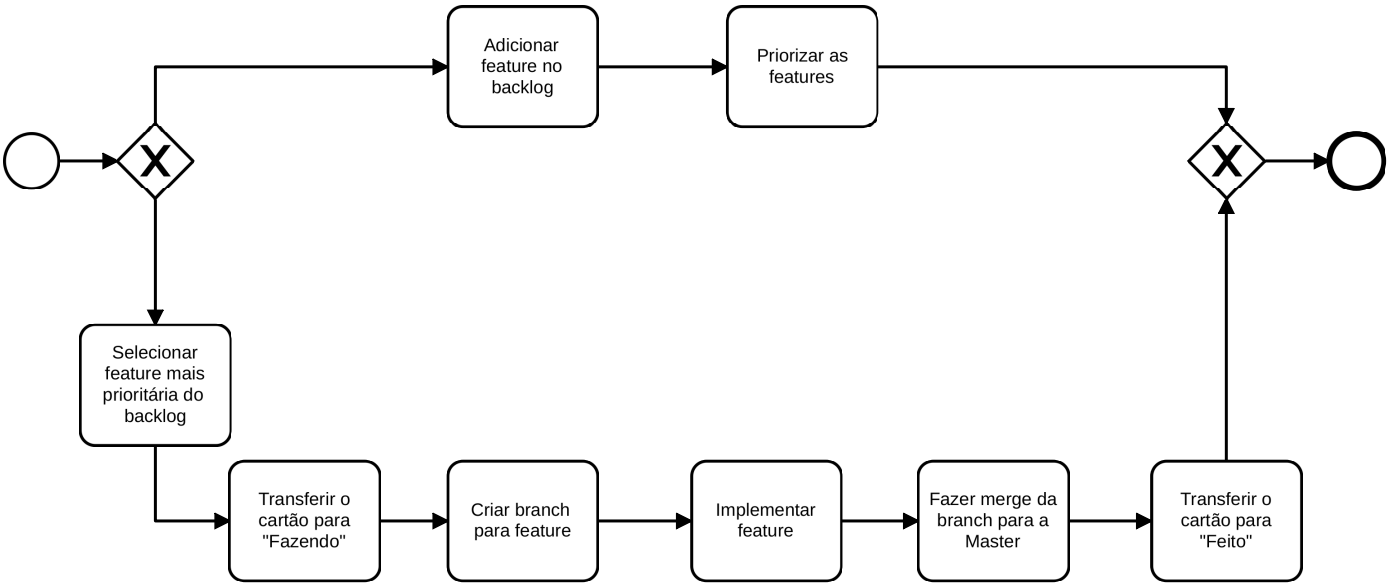


Figura 3 – Processo de desenvolvimento.

3.2 Requisitos

3.2.1 Requisitos Funcionais

Os requisitos funcionais foram definidos a partir do problema encontrado e da justificativa proposta. Ou seja, os requisitos estão intrinsecamente relacionados com a necessidade de alertar agentes da saúde, em qualquer localidade, acerca de reações adversas e alérgicas de uma pessoa e ao mesmo tempo fornecer um ambiente em que o usuário, que é dono dessas informações, é responsável por fornecê-las.

Requisito Funcional	Decrição
RF1	Permitir o cadastro de medicamentos e princípios ativos
RF2	Permitir o cadastro de hospitais e médicos
RF3	Permitir o cadastro de administradores de hospitais e administradores do sistema
RF4	Permitir o cadastro de usuários
RF5	Permitir o cadastro de uma reação adversa de um usuário
RF6	Permitir verificar se um medicamento é seguro para consumo para um usuário
RF7	Permitir que um profissional da área de saúde recupere as reações adversas de um usuário, caso seja necessário
RF8	Permitir que um profissional da área de saúde atualize as informações de reações adversas de um usuário, caso seja necessário
RF9	Permitir que um usuário seja notificado caso algum agente de saúde acesse suas informações

Tabela 8: Requisitos funcionais.

3.2.2 Requisitos Não Funcionais

3.2.2.1 Segurança

Por tratar-se de informações de saúde, é necessário que toda a manipulação e utilização das mesmas seja realizada por usuários autenticados na aplicação.

Além disso, a comunicação do usuário com o servidor deve ser sigilosa, no sentido que nenhuma pessoa deve ser capaz de visualizar os dados trafegando na comunicação, e o servidor deve ter certeza de que a requisição não foi alterada, conferindo a integridade da requisição.

Outros critérios de segurança foram retirados da Sociedade Brasileira de Informática em Saúde (SBIS). O órgão oferece um certificado para sistemas de registro eletrônico de saúde, caso os mesmos consigam atender aos critérios, havendo dois níveis de certificação (Sociedade Brasileira de Informática em Saúde, 2013):

NGS1: Adequada para sistemas que coexistem com a impressão de documentos médicos;

NGS2: Adequada para sistemas que podem suprimir a impressão de documentos médicos;

O sistema não possui o objetivo de eliminar os documentos em papel, almejando auxiliar no tratamento médico apenas com a disponibilização de informações, que poderão ser utilizadas inclusive para impressão nas unidades hospitalares.

Portanto, escolheu-se os critérios de segurança considerados adequados para o projeto, considerando as limitações de tempo e escopo, à partir dos critérios presentes no

NGS1. Dessa forma, foram eliminados tópicos que não conseguiriam agregar demasiado valor à solução proposta.

Como exemplo de um critério eliminado, pode-se citar a implementação de gerenciador de espaço de armazenamento, para que não seja possível inserir mais registros após um limiar. Além disso, um profissional de TI deve ser notificado, caso esse limiar seja alcançado. Esse critério possui o identificador NGS1.05.03 (Sociedade Brasileira de Informática em Saúde, 2013).

Os critérios escolhidos são listados abaixo, sendo que sua definição formal pode ser encontrada no documento fornecido por (Sociedade Brasileira de Informática em Saúde, 2013):

1. 1.02.02
2. 1.02.03
3. 1.02.04
4. 1.02.05
5. 1.02.06
6. 1.02.08
7. 1.02.09
8. 1.03.01
9. 1.03.02
10. 1.03.03
11. 1.04.01
12. 1.04.12
13. 1.05.01
14. 1.05.02
15. 1.06.01
16. 1.06.03
17. 1.07.04
18. 1.07.05

19. 1.09.01

20. 1.09.02

21. 1.09.12

22. 1.11.01

23. 1.12.01

24. 1.12.02

É importante notar que esses critérios orientaram, de forma geral, o desenvolvimento da segurança da aplicação. Porém, podem ter sido interpretados ou implementados de uma forma diferente.

O resultado da implementação dos critérios é descrito nos resultados do trabalho.

3.2.2.2 Diagrama de Caso de Uso

Com os requisitos definidos, pode-se traçar um diagrama que permite a visualização de quais são as funcionalidades que devem ser oferecidas e para quais atores, que se dividem nos respectivos papéis:

- *User*: Inserir as informações acerca de suas reações adversas e medicamentos utilizados;

- *Carer*: Visualizar as reações de um usuário na hora do atendimento médico, de forma a evitar um acidente. Além disso, pode inserir uma reação que observou durante o tratamento do paciente;

- *Sysadmin*: Inserir os hospitais, remédios e princípios ativos na aplicação. Idealmente, é um funcionário do Ministério da Saúde com contatos na ANVISA;

- *Admin*: É o diretor de um hospital, responsável por aceitar outros usuários como médicos associados ao hospital que gerencia;

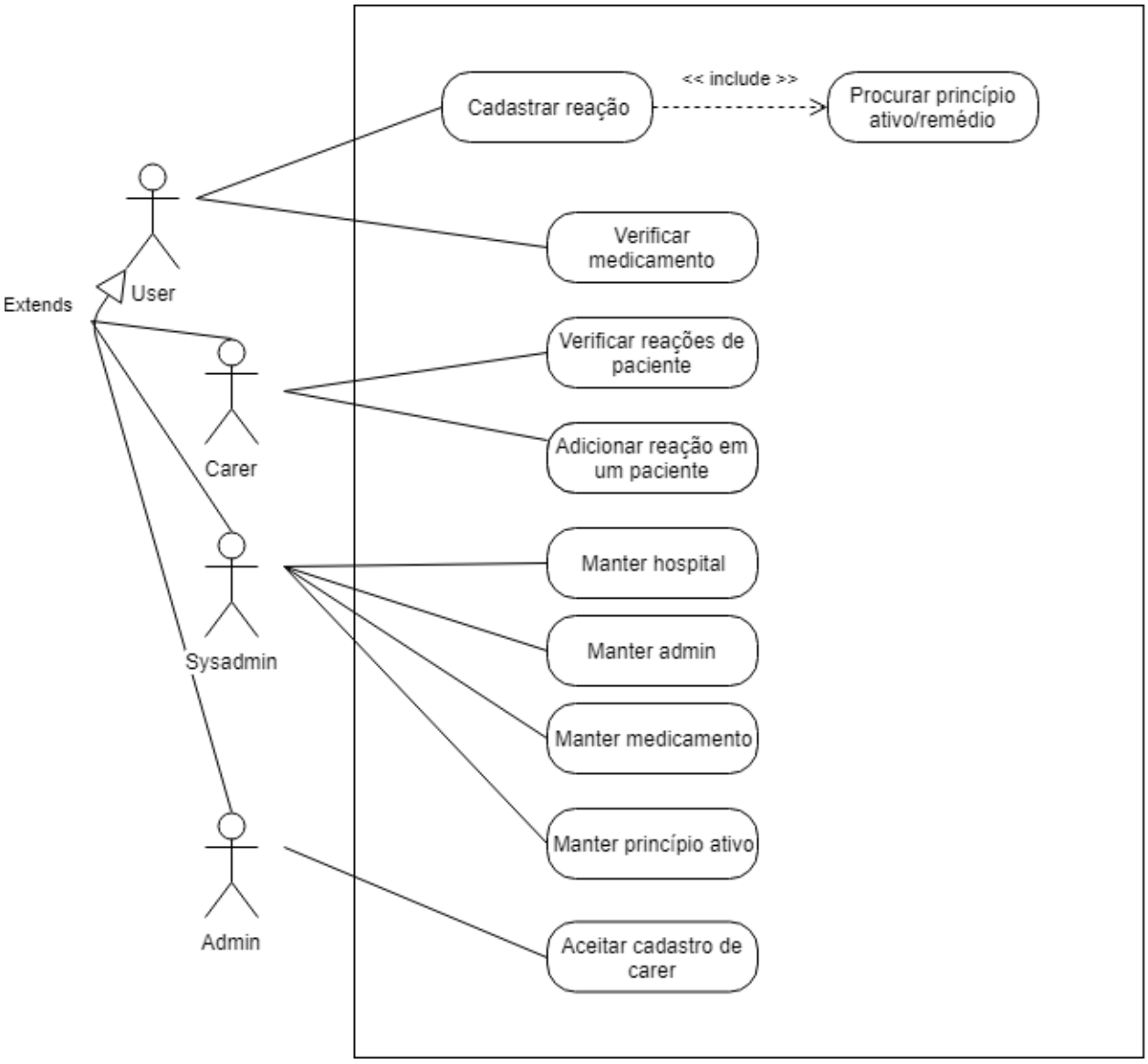


Figura 4 – Diagrama de Caso de Uso do sistema.

É importante notar que a nomenclatura de herança foi utilizada no diagrama acima para especificar os tipos diferentes de usuários e seus respectivos poderes dentro da aplicação.

Porém, a aplicação trabalhará com o conceito de papéis que um usuário pode desempenhar. Esses papéis possuem suas respectivas funcionalidades e um usuário pode desempenhar qualquer quantidade de papéis, de todos a nenhum.

Em outras palavras, pode haver um usuário que é, ao mesmo tempo, *admin*, *sysadmin* e *carer* e que portanto terá acesso a todas as funcionalidades da aplicação. Da mesma forma, pode haver um usuário que não desempenha papel algum.

3.3 Solução Proposta

A solução proposta é uma forma de centralizar informações relativas à reações adversas e alérgicas de um indivíduo, além de fornecer ao mesmo uma forma de gerenciar essas informações e verificar no cotidiano a segurança do uso de diferentes medicamentos.

Para alcançar esse objetivo, a solução proposta baseia-se na construção de dois elementos primários:

3.3.1 API

Responsável por armazenar as informações de um indivíduo, salvando-as e fornecendo-as à qualquer médico ou enfermeiro que necessite visualizá-las.

É importante ressaltar que por ser um sistema de registro pessoal de saúde, todas as informações disponibilizadas na API devem ser cadastradas pelo próprio indivíduo, que terá total ciência da possibilidade de profissionais da área médica visualizarem as mesmas.

3.3.2 Interface *Web*

Agrega as funções de gerenciamento de alergias e de verificação da segurança de consumo de um medicamento. Na interface, o usuário poderá:

- Inserir informações sobre suas reações adversas;
- Verificar através do princípio ativo de um medicamento se o mesmo é seguro para consumo;
- Requisitar informações sobre reações adversas de outros usuários (dado que o usuário utilizando seja da área médica)

As funcionalidades descritas acima ocorrem através da interação da interface com a API, requisitando e atualizando as informações presentes na mesma. Portanto, a página *web* é, também, um modelo de cliente para o consumo da API.

Porém, não deve ser a única forma de recuperar ou atualizar as informações da mesma. A idéia por trás de disponibilizar informações tão relevantes em uma interface é que qualquer sistema médico possa utilizá-la com poucas modificações. Esse cenário pode ser visto na figura abaixo:

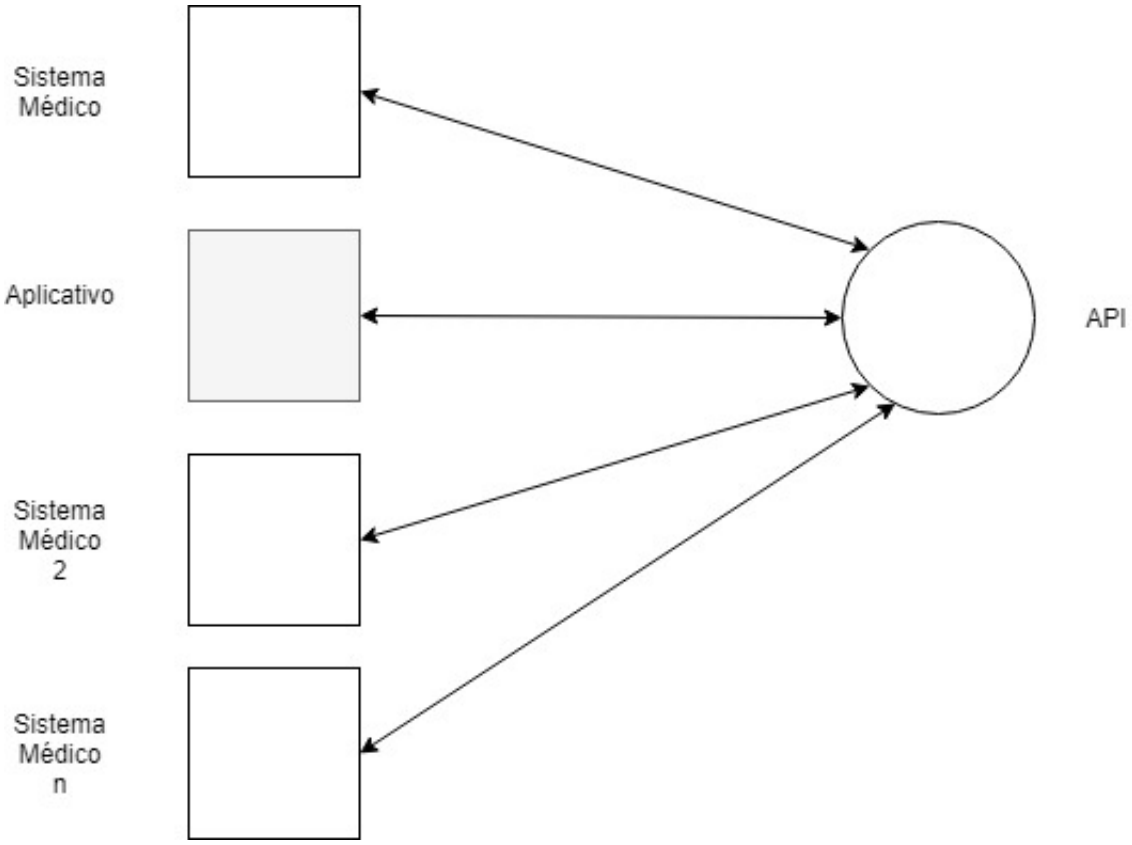


Figura 5 – Diferentes sistemas interagindo com a API.

3.4 Desenvolvimento da Solução

3.4.1 Ferramentas

3.4.1.1 GitLab

GitLab é uma ferramenta que oferece funcionalidades acerca de diversos tópicos diferentes, dentre os quais serão úteis para o projeto:

Gerenciamento de código: É possível criar repositórios (GITLAB, 2018a) privados de forma gratuita na plataforma do GitLab. O projeto conta com dois repositórios, que são correspondentes a API e à interface *web*.

CI/CD: Há uma ferramenta nativa no GitLab para integração e *deploy* contínuos, sendo que a configuração da mesma ocorre através de um arquivo disponível na raíz do projeto (GITLAB, 2018b). A ferramenta será utilizada para garantir que todos os *merge requests*, contendo novas funcionalidades, para a *branch master* do projeto façam com que um *pipeline* seja acionado com o fim de garantir a integridade do software com aquelas funcionalidades.

Planejamento do projeto: Há uma funcionalidade que permite a criação de um quadro com as *issues* do projeto (GITLAB, 2018d). Esse quadro pode ser adaptado para diversos fluxos de trabalhos diferentes (GITLAB, 2018d), sendo compatível com o requerimento de uso da metodologia Kanban. Haverão três colunas no quadro: À Fazer, Fazendo e Feito.

O endereço dos repositórios contendo os códigos da API e da interface *web* são, respectivamente:

- https://gitlab.com/navarrovmn/alergio-api
- https://gitlab.com/navarrovmn/tcc-frontend

3.4.1.2 Ruby on Rails

Rails é um *framework open-source* para desenvolvimento *web* que utiliza a linguagem Ruby e possui diversas facilidades para a produção de *software* dessa categoria. Baseado no padrão *Model-View-Controller*, tem dois princípios básicos (RAILS, 2018):

***Don't Repeat Yourself*:** Não deve haver repetição de código, pois isso dificulta a manutenibilidade do projeto. Se uma mesma lógica aparece em diversos pontos do código, ela deve ser modularizada para evitar replicação de código.

***Convention Over Configuration*:** Esse aspecto diz que os procedimentos devem ser realizados do jeito mais usual do *framework*, sem grandes alterações sem necessidade. Isso significa, na prática, sempre saber onde estão certos arquivos, lógicas e estruturas, pois a convenção diz onde se encontram ou como são feitos.

O *framework* será utilizado para a construção da API. Por requisitos de segurança da (Sociedade Brasileira de Informática em Saúde, 2013), todo o processamento deve ser realizado no servidor. Isso implica que todo o processamento será feito com Rails no *backend*, ou seja, na API.

3.4.1.3 Vue.js

Vue.js é um *framework* JavaScript para produção de interfaces de usuário com desenvolvimento baseado na criação e reutilização de componentes, sendo possível encontrar diversas bibliotecas de componentes prontos para uso (INTRODUCTION, 2018).

Neste projeto, será utilizado o *framework* Vuetify, que oferece componentes inspirados no Material Design, além de permitir a criação de um *Progressive Web App* (PWA) de forma simplificada (MATERIAL..., 2018).

A interface construída é um exemplo de exploração das capacidades da API, fornecendo um fluxo básico da aplicação.

3.4.1.4 JSON Web Token

JWT é um mecanismo de segurança que utiliza um objeto JSON para fornecer segurança no compartilhamento de dados entre duas entidades (INTRODUCTION..., 2018).

Ele será utilizado no projeto no âmbito de autenticação, sendo seu fluxo descrito abaixo (INTRODUCTION..., 2018):

1. Usuário informa suas credenciais (usuário e senha) para o servidor;
2. O servidor confere as credenciais e caso tudo esteja correto, cria um *token* para o usuário;
3. Para cada requisição que o usuário fizer de agora em diante, ele deve incluir o *token* no cabeçalho do pacote;
4. O servidor recebe a requisição e confere se o *token* é válido ou não antes de processar a requisição;

Uma observação importante em relação ao JWT é o fato de ele ser *stateless*, visto que o servidor não mantém informações sobre o usuário em memória, como é realizado utilizando sessões (INTRODUCTION..., 2018).

3.4.1.5 Amazon AWS Elastic Beanstalk

Serviço *cloud* para implantação de aplicativos oferecendo a vantagem de aliviar o desenvolvedor da configuração de aspectos de infraestrutura (AMAZON..., 2018).

Outra vantagem da utilização desse serviço é a escalabilidade proporcionada pelo mesmo de acordo com o crescimento e necessidade da aplicação, além de contar também com balanceamento de carga, por exemplo (AMAZON..., 2018). A API será implantada neste serviço para tirar proveito de todas as funcionalidades supracitadas.

3.4.1.6 Docker

Docker é uma alternativa ao uso de *Virtual Machines*, utilizando uma ideia similar conhecida como *container*, que isola um conjunto de pacotes, bibliotecas e *softwares* para fornecer um serviço (WHAT..., 2018).

Todos os *containers* que executam em uma máquina são processos no espaço de usuário que estão sobre o motor do Docker. Além disso, os mesmos compartilham o mesmo *kernel* (WHAT..., 2018).

Há diversas possibilidades de uso com o Docker. Apesar de ser uma alternativa a VM no sentido de isolamento e criação de ambientes, ele funciona bem em uma VM. Ou seja, pode-se instalar o docker em uma máquina virtual e nela executar os *containers* necessários para o serviço.

A API do projeto será oferecida à comunidade em *containers* ao fim do projeto, junto com o código disponibilizado no GitLab.

3.4.2 Fonte de Dados

Há duas fontes de dados que alimentam o sistema:

3.4.2.1 Usuário

Responsável por inserir as informações pessoais de reações adversas que serão consultadas pelos profissionais de saúde.

Essas informações também serão utilizadas para verificação de segurança no consumo de medicamentos por parte do usuário.

3.4.2.2 Agência Nacional de Vigilância Sanitária

Através do Portal Brasileiro de Dados Abertos (<http://dados.gov.br/>), a ANVISA liberou informações acerca dos remédios cadastrados em sua base de dados. Essas informações estão disponibilizadas em CSV no seguinte *link*:

- <http://dados.gov.br/dataset/anvisa-precos-de-medicamentos>

Esse CSV possui as seguintes colunas de interesse para o projeto, de acordo com o dicionário de dados disponibilizado no endereço supracitado:

NU_REGISTRO: Registro na Anvisa;

CO_EAN: Código de barras do remédio;

NO_PRODUTO: Nome do remédio;

DS_SUBSTANCIA: Substância do remédio;

Além desse arquivo, a Anvisa disponibiliza em seu portal <http://portal.anvisa.gov.br/> a lista de medicamentos de referência, similares e genéricos nos seguintes *links*:

- <http://portal.anvisa.gov.br/registros-e-autorizacoes/medicamentos/produtos/medicamentos-de-referencia/lista>
- <http://portal.anvisa.gov.br/documents/219201/219401/Lista%2Bsite%2B26-01-16%2BEXCEL.pdf/b5e6d58d-5315-4a73-b1ed-25e289d1e2f5>
- <http://portal.anvisa.gov.br/medicamentos-genericos-registrados>

Essas listas encontram-se em PDF, o que torna o consumo de suas informações de forma automatizada mais problemático. Entretanto, depois de entrar em contato com a Anvisa, as mesmas listas foram fornecidas em XLSX. Para a construção da base de dados de medicamentos da aplicação, será utilizado informações tanto do CSV disponibilizado no Portal de Dados Abertos quanto da lista de medicamentos de referência disponibilizada no primeiro endereço da lista acima.

3.4.3 Resultados Esperados

Como resultado esperado, tem-se a elaboração de um sistema que seja efetivo em informar agentes de saúde sobre os dados médicos de um paciente, com o intuito de evitar e portanto reduzir os casos de incidentes relacionados à reações medicamentosas.

Esse sistema deve contemplar os casos de uso supracitados e atender à arquitetura proposta no diagrama de classes, bem como utilizar efetivamente as tecnologias citadas para seus respectivos propósitos.

4 Resultados Obtidos

Nesse capítulo, serão explicados os resultados obtidos pelo desenvolvimento da solução proposta. O capítulo está dividido em três seções:

- *Continuous Integration/Continuous Deploy*: será detalhado o processo de desenvolvimento elaborado para o projeto;
- API: será explicado o que foi alcançado, especialmente em relação aos critérios de segurança;
- Interface *Web*: detalhamento do *front-end* desenvolvido;
- Próximos Passos: planejamento para o futuro do *software*;

4.1 *Continuous Integration/Continuous Deploy*

O CI/CD do projeto utilizou a ferramenta nativa do GitLab, que permite realizar diversas ações toda vez que uma alteração no código é detectada. Essas ações são divididas em *stages*, que podem conter um ou mais *jobs* (GITLAB, 2018c).

De uma maneira geral, o *job* é o que executa as ações, ocorrendo de forma paralela a todos os outros do mesmo *stage*. Esse último, por sua vez, ocorre de forma sequencial. Tudo isso é definido em um arquivo nomeado .gitlab-ci.yml na raíz do projeto (GITLAB, 2018c). O projeto conta com dois *stages*:

- *Test*;
- *Deploy*;

No *stage* de teste, os testes funcionais e unitários da aplicação são executados, verificando a integridade do sistema. Além disso, há uma análise estática da qualidade de código também, baseada no motor do *Code Climate*.

No *stage* de *deploy*, que ocorre somente se os testes executarem sem erros, a aplicação é atualizada pelo AWS Elastic Beanstalk.

Com a explicação acima, pode-se explicar o processo de desenvolvimento utilizado na solução e ilustrado pela Figura 6.

A *branch* base para desenvolvimento é a *devel*, que deve estar sempre sincronizada ou à frente da *master*.As *branches* de desenvolvimento de funcionalidades baseiam-se, então, na *devel*.

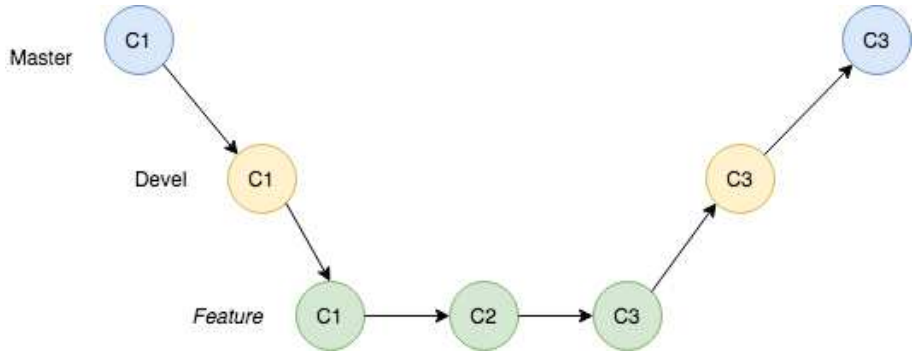


Figura 6 – Processo de Desenvolvimento.

Toda vez que um *commit* é enviado para o repositório em qualquer *branch*, ele ativa o *pipeline* configurado no projeto. Porém, em todas as *branches* exceto a *master*, ele executará apenas os testes e análise estática de código.

Após o desenvolvimento da funcionalidade, abre-se um *merge request* para a *devel*. Com uma ou mais funcionalidades contidas na *devel*, pode-se atualizar a *master* através de outro MR.

Quando a *master* é atualizada, os testes são executados normalmente. Porém, ao fim e sucesso dos testes, o *stage* de *deploy* é acionado, informando ao Beanstalk para atualizar a versão em produção com as novas mudanças.

4.2 API

A API do projeto foi desenvolvida utilizando o *framework* Ruby on Rails na versão 5.2.1. A modelagem do banco de dados está representada na Figura 7.

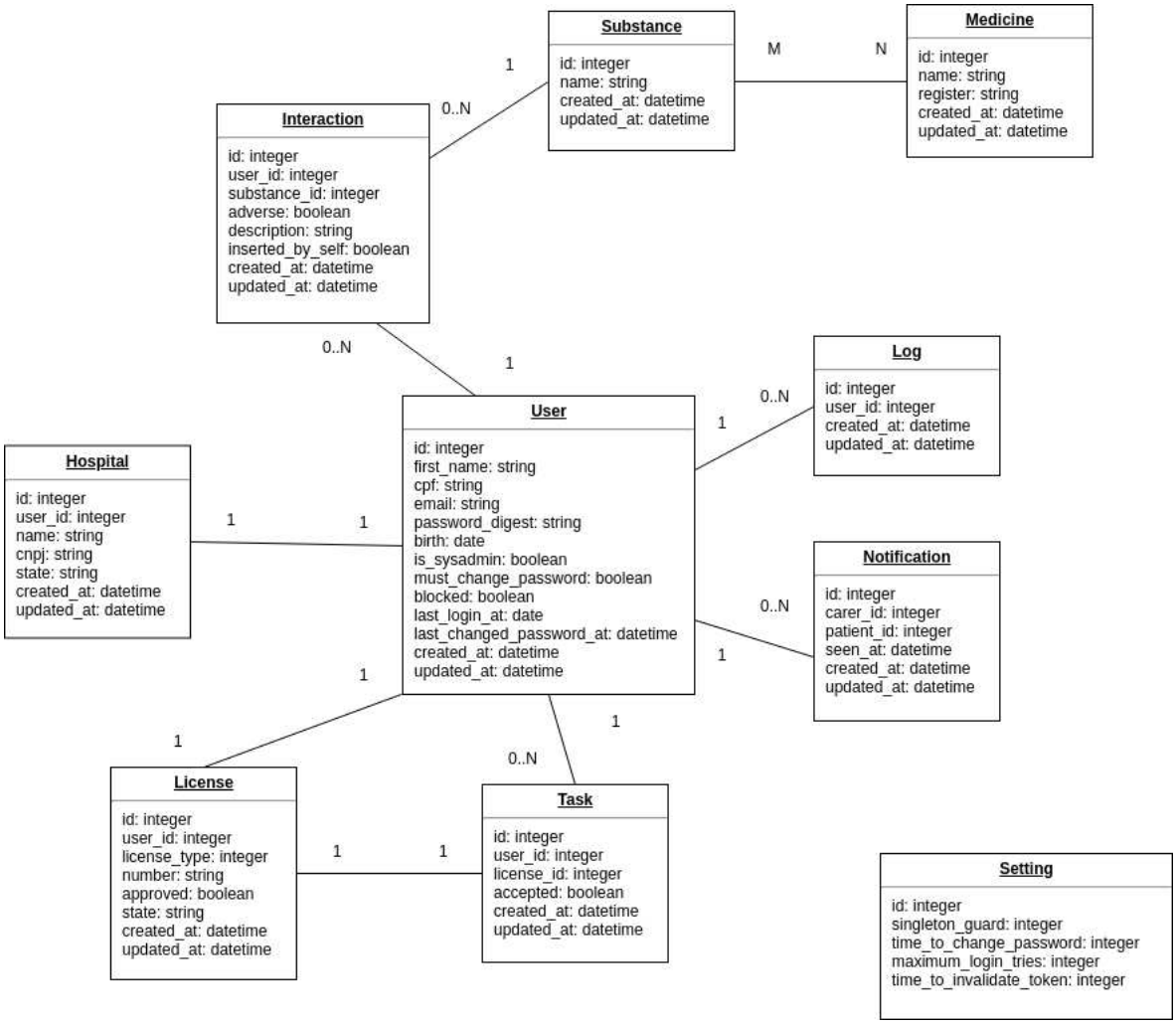


Figura 7 – Modelagem do Banco de Dados.

- *User*: Representa um usuário, apresentando um conjunto de atributos para identificação e outros atributos para contemplar os requisitos de segurança, dentre eles qual a última vez que a senha foi trocada ou se ele foi bloqueado;
- *Medicine*: Representa um medicamento, que pode possuir vários princípios ativos;
- *Substance*: É a representação de um princípio ativo, estando em vários medicamentos;
- *Interaction*: Representa uma reação de um usuário. É a tabela associativa de um usuário com substância, descrita acima. Através do atributo *adverse*, pode-se tipificar uma reação como sendo adversa ou como sendo de consumo, para que um usuário possa cadastrar quais princípios ativos ele está atualmente consumindo;
- *Log*: Tabela responsável por arquivar todas as tentativas não sucedidas de *login* de um usuário, para manter um registro dessas ocorrências e verificar se ele deve ser bloqueado ou não;

- *Notification*: Tabela responsável por arquivar as consultas realizadas por agentes de saúde à determinadas pessoas. Através dessa tabela, é possível notificar a um usuário que suas informações foram pesquisadas por um agente e qual agente. Ao mesmo tempo, é possível verificar todas as consultas realizadas por um agente;

- *Hospital*: Tabela que representa um hospital. A função principal dessa tabela é mapear um usuário como diretor de um hospital. Dessa forma, todo usuário que quiser tornar-se um agente de saúde na plataforma, deve-se associar à um hospital. Isso necessitará da aprovação do diretor daquele hospital em reconhecer que o usuário é, de fato, um agente de saúde;

- *License*: Entidade que mapeia uma licença para um usuário. Através do atributo *license_type*, pode-se cadastrar um CRM ou um COREN, para diferentes agentes de saúde. Ao criar uma licença, o diretor do hospital associado é notificado e deve aprovar esse cadastro, tornando a licença válida. Um usuário é reconhecido como um agente de saúde na aplicação quando possui uma licença aprovada;

- *Task*: Entidade que representa a notificação para o diretor de um hospital que deve ser aprovada ou rejeitada para aceitar um usuário como agente de saúde dentro da plataforma;

- *Setting*: Entidade responsável por guardar atributos acerca do funcionamento geral da aplicação. Guarda três variáveis: duração do tempo de um token, número máximo de tentativas de *login* e tempo obrigatório para troca de senhas;

Há também dois módulos ausentes no banco que são imprescindíveis para a aplicação:

- *Authentication*: Responsável por controlar o acesso aos *endpoints* da API. É nesse módulo que é gerado o *token* de acesso, bem como a verificação se um *token* fornecido é válido ou não, se o usuário deve mudar a senha ou mesmo se o usuário deve ser bloqueado;

- *Permission*: Ocorre após a autenticação e é responsável por verificar se o usuário pode acessar o *endpoint* requisitado, dado o nível de acesso do mesmo dentro da aplicação.

De uma forma geral, um usuário não pode ver nenhuma informação de outro usuário, salvo duas exceções:

- Se o usuário requisitando a ação é um agente de saúde e a ação requisitada consta em suas ações permitidas;

- Se o usuário requisitando a ação é um *sysadmin* e a ação requisitada consta em suas ações permitidas;

  As ações permitidas para um agente de saúde são:

- Verificar as reações de um usuário;
- Criar uma reação para um usuário;
- Verificar as informações de um usuário;

  As ações permitidas para um *sysadmin* são:

- Desbloquear um usuário;
- Pedir a troca de senha de um usuário;
- Verificar as informações de um usuário;

4.2.1 Critérios de Segurança

O desenvolvimento da API seguiu com os critérios de segurança como principal foco de implementação. Nesse contexto, foram uma orientação para a implementação das funcionalidades.

Isso significa que os critérios eram lidos e interpretados e adaptados ao problema da melhor forma possível, caso o critério não fosse explícito em sua requisição. Portanto, alguns critérios necessitam de uma explicação mais detalhada, sendo eles:

- NGS 1.02.03;
- NGS1.03.01;
- NGS1.05.01;
- NGS1.05.02;
- NGS1.07.07;
- NGS1.07.10;
- NGS1.09.04;
- NGS1.12.01;
- NGS1.12.02;

- NGS1.12.03;

Os outros critérios foram implementados sem maiores ressalvas.

4.2.1.1 NGS 1.02.03

A senha do usuário é armazenada no banco utilizando *Bcrypt*, um método de *hash* que aceita uma variável de custo para informar quanto poder de processamento deve ser utilizado para realizar o *hash* da senha.

4.2.1.2 NGS1.03.01

Ao utilizar o *JWT* para autenticação, pode-se determinar um tempo de validade para o *token*. Dessa forma, após esse tempo, qualquer operação utilizando aquele *token* não será mais autorizada.

Esse critério foi contemplado da forma descrita acima, sendo que o tempo de validade de um *token* é configurável na API. Uma desvantagem dessa forma de implementação é que a atividade do usuário está limitada ao tempo configurado pelo administrador.

4.2.1.3 NGS1.05.01, NGS1.05.02 e NGS1.07.07

Estes critérios de segurança foram complicados de contemplar em um projeto de *software* livre e de uma API, em que é proposto que a solução seja adotada por outras entidades em outros locais e sempre melhorada pela comunidade.

Esta implementação da API beneficiou-se do *deploy* contínuo na AWS para garantir *backups* diários e íntegros. Porém, outros projetos que utilizem a solução descrita aqui devem certificar que esses critérios sejam atendidos na implementação dos mesmos.

De forma análoga, é papel do implementador que essas informações só sejam acessíveis por pessoas de confiança do projeto, para que a reconstrução desse banco seja impossível pelas mãos de alguém não associado à comunidade.

4.2.1.4 NGS1.07.10

Os dados são validados ao chegarem na API, para que estejam preenchidos ou tenham o formato e tamanho corretos. Além disso, são utilizados métodos do ORM do Rails para fazer requisições ao banco, diminuindo a possibilidade de ataque por *SQL Injection*.

Porém, os cuidados descritos no critério que contemplam um front-end não entram na proposta do desenvolvimento, visto que essas interfaces devem ser desenvolvidas por terceiros. O cuidado tomado é que a API esteja protegida independente de onde vem a requisição.

4.2.1.5 NGS1.09.04

Este critério também enquadra-se na explicação supracitada. Instruir como gerar um *backup* depende de diversas variáveis que não podem ser todas atendidas, como qual banco é utilizado e onde e como foi realizado o *deploy*. Nesse projeto, o backup é gerado automaticamente todos os dias pelo serviço de banco da AWS.

4.2.1.6 NGS1.12.01 e NGS1.12.02

No contexto deste projeto, não é interessante adotar uma solução que informe ao usuário para quê os dados serão usados após o cadastro do mesmo.

Portanto, a solução adotada inclui deixar esse termo de consentimento no *front-end*, na página de cadastro. Toda interface que desejar se comunicar com a API deve implementar este aviso antes do cadastro.

Dessa forma, o próprio ato de estar cadastrado já garante que o usuário consentiu com o uso das informações dele com o propósito de tornar acessível suas reações adversas para diversos hospitais e médicos pelo Brasil.

A desvantagem desta solução é a de que deve-se fiscalizar as interfaces que se comunicam com o *backend* com o objetivo de verificar que as mesmas cumprem esse critério.

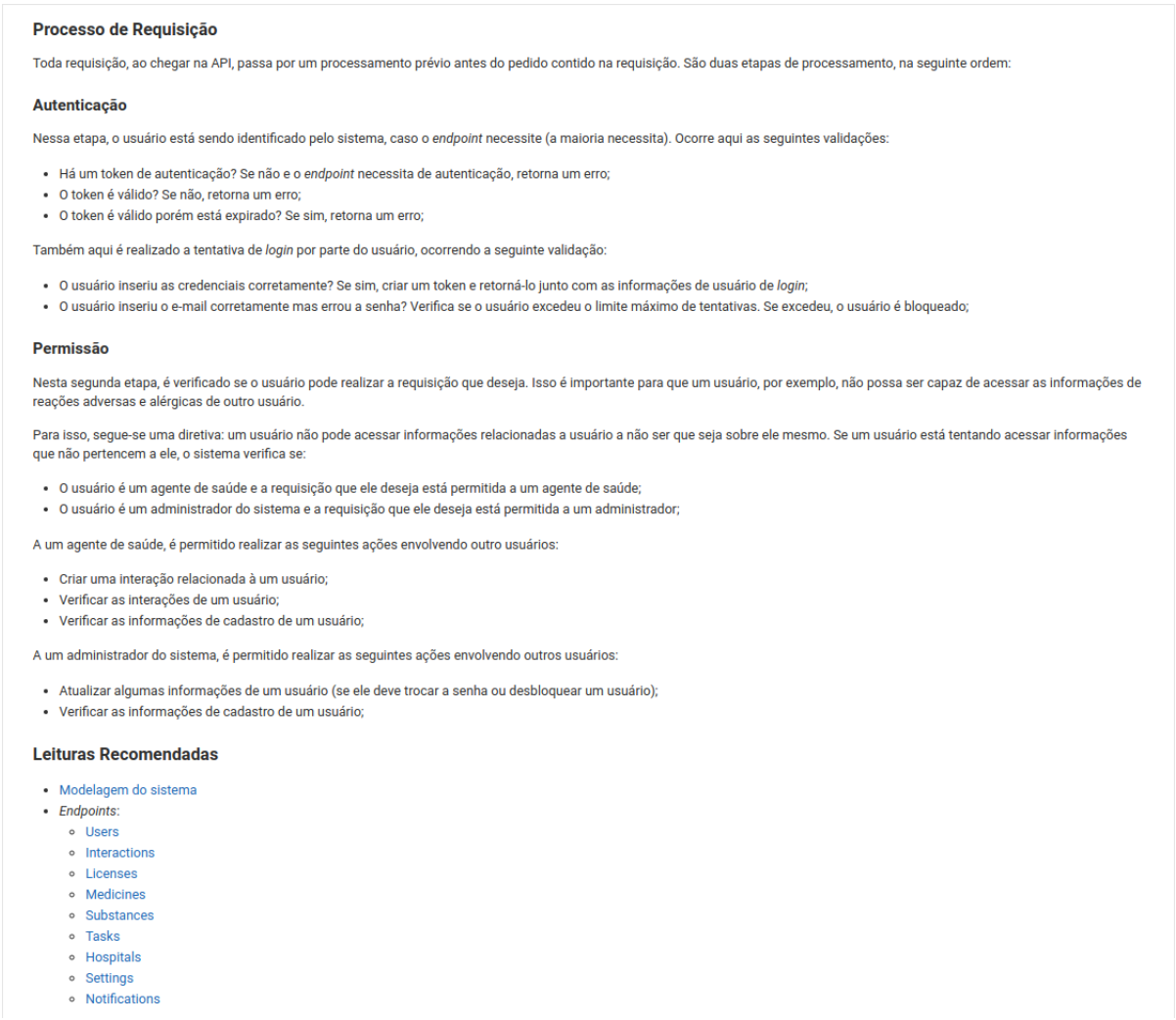
4.2.1.7 NGS1.12.03

Por entender que o usuário cadastrado já entende como serão utilizadas as suas informações e que as interfaces que se comunicam com a API também entendem o propósito e os dados disponíveis, esse critério tornou-se desnecessário, não sendo contemplado pelo projeto.

4.2.2 Documentação

A API está documentada em seu repositório, contendo desde modelagem do banco com explicação para cada entidade até a documentação dos *endpoints* disponíveis, com formato de mensagem e argumentos esperados.

Esse aspecto é de suma importância em projetos de *software* livre, para que o aprendizado leve o menor tempo possível e, portanto, a contribuição de agentes externos possa vir de forma mais eficaz e rápida.



Processo de Requisição

Toda requisição, ao chegar na API, passa por um processamento prévio antes do pedido contido na requisição. São duas etapas de processamento, na seguinte ordem:

Autenticação

Nessa etapa, o usuário está sendo identificado pelo sistema, caso o *endpoint* necessite (a maioria necessita). Ocorre aqui as seguintes validações:

- Há um token de autenticação? Se não e o *endpoint* necessita de autenticação, retorna um erro;
- O token é válido? Se não, retorna um erro;
- O token é válido porém está expirado? Se sim, retorna um erro;

Também aqui é realizado a tentativa de *login* por parte do usuário, ocorrendo a seguinte validação:

- O usuário inseriu as credenciais corretamente? Se sim, criar um token e retorná-lo junto com as informações de usuário de *login*;
- O usuário inseriu o e-mail corretamente mas errou a senha? Verifica se o usuário excedeu o limite máximo de tentativas. Se excedeu, o usuário é bloqueado;

Permissão

Nesta segunda etapa, é verificado se o usuário pode realizar a requisição que deseja. Isso é importante para que um usuário, por exemplo, não possa ser capaz de acessar as informações de reações adversas e alérgicas de outro usuário.

Para isso, segue-se uma diretiva: um usuário não pode acessar informações relacionadas a usuário a não ser que seja sobre ele mesmo. Se um usuário está tentando acessar informações que não pertencem a ele, o sistema verifica se:

- O usuário é um agente de saúde e a requisição que ele deseja está permitida a um agente de saúde;
- O usuário é um administrador do sistema e a requisição que ele deseja está permitida a um administrador;

A um agente de saúde, é permitido realizar as seguintes ações envolvendo outro usuários:

- Criar uma interação relacionada à um usuário;
- Verificar as interações de um usuário;
- Verificar as informações de cadastro de um usuário;

A um administrador do sistema, é permitido realizar as seguintes ações envolvendo outros usuários:

- Atualizar algumas informações de um usuário (se ele deve trocar a senha ou desbloquear um usuário);
- Verificar as informações de cadastro de um usuário;

Leituras Recomendadas

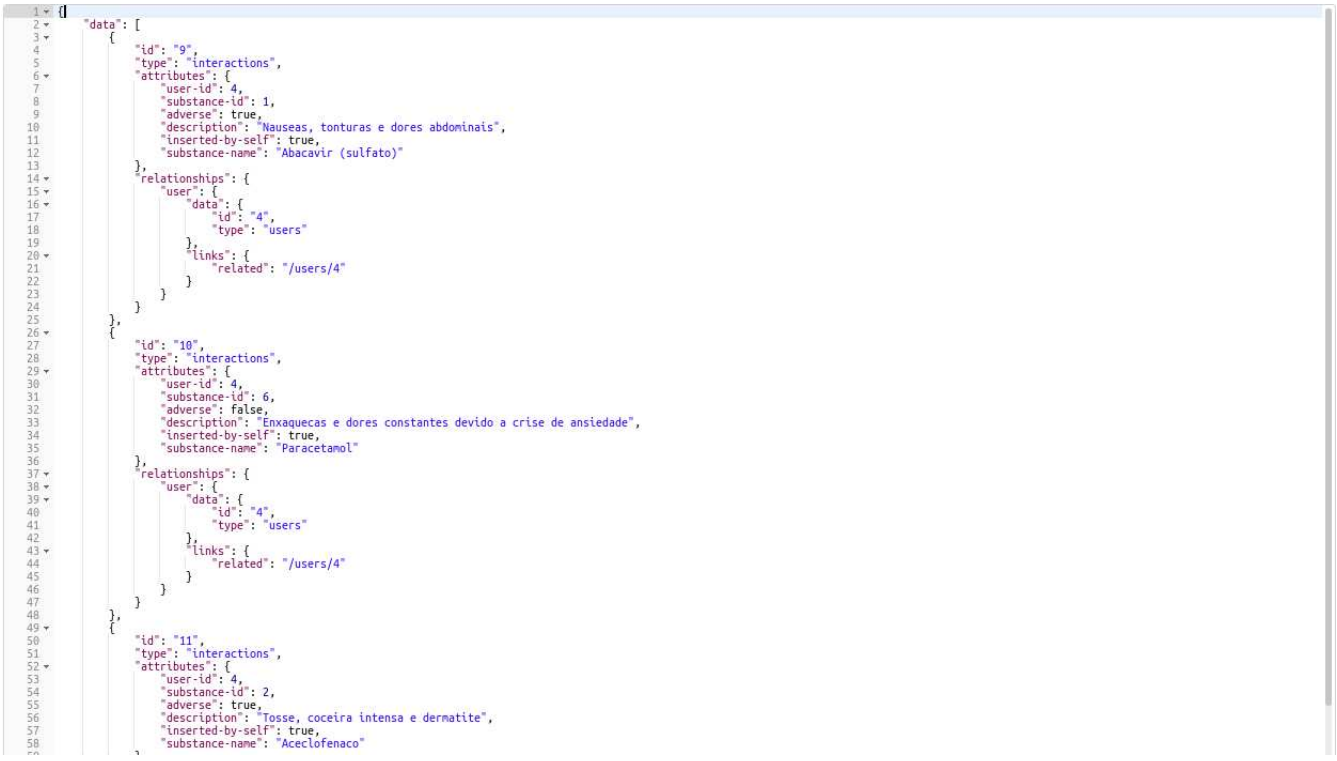
- Modelagem do sistema
- *Endpoints*:
  - Users
  - Interactions
  - Licenses
  - Medicines
  - Substances
  - Tasks
  - Hospitals
  - Settings
  - Notifications

Figura 8 – Página de documentação do Allergio.

Como exemplo de requisições, pode-se verificar na Figura 9 o retorno de um *login* na aplicação, seguido de uma requisição para recuperar as reações adversas e medicamentos consumidos do usuário na Figura 10.


```
{
    "data": {
        "id": "4",
        "type": "users",
        "attributes": {
            "first-name": "Victor",
            "birth": "1996-03-31",
            "last-login-at": "2019-03-24",
            "is-sysadmin": false,
            "is-medic": false,
            "logs": [
                {
                    "id": 1,
                    "user_id": 4,
                    "created_at": "2019-03-24T14:41:00.902Z",
                    "updated_at": "2019-03-24T14:41:00.902Z"
                }
            ],
            "token": "eyJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2lkIjo0LCJleHAiOjE1NTM0Mzg3NzJ9.oTpSFb9Ki_U9sL8sK5CjFXyojTu4za-hHToHpX6Goog"
        }
    }
}
```


Figura 9 – Resposta para uma requisição de *login*.



```
{
  "data": [
    {
      "id": "9",
      "type": "interactions",
      "attributes": {
        "user-id": 4,
        "substance-id": 1,
        "adverse": true,
        "description": "Nauseas, tonturas e dores abdominais",
        "inserted-by-self": true,
        "substance-name": "Abacavir (sulfato)"
      },
      "relationships": {
        "user": {
          "data": {
            "id": "4",
            "type": "users"
          },
          "links": {
            "related": "/users/4"
          }
        }
      }
    },
    {
      "id": "10",
      "type": "interactions",
      "attributes": {
        "user-id": 4,
        "substance-id": 6,
        "adverse": false,
        "description": "Enxaquecas e dores constantes devido a crise de ansiedade",
        "inserted-by-self": true,
        "substance-name": "Paracetamol"
      },
      "relationships": {
        "user": {
          "data": {
            "id": "4",
            "type": "users"
          },
          "links": {
            "related": "/users/4"
          }
        }
      }
    },
    {
      "id": "11",
      "type": "interactions",
      "attributes": {
        "user-id": 4,
        "substance-id": 2,
        "adverse": true,
        "description": "Tosse, coceira intensa e dermatite",
        "inserted-by-self": true,
        "substance-name": "Aceclofenaco"
```


Figura 10 – Resposta para uma requisição de recuperação de reações adversas.

4.3 Interface *Web*

A *interface* para consumir a API foi construída utilizando Vue.js. Além disso, utilizou-se o *framework* Vuetify para o uso de componentes que seguem a identidade visual do *Material Design*.

O *front-end* foi construído como um PWA, sendo possível portanto adicionar a página em um dispositivo móvel para utilização como se fosse um aplicativo comum. Para isso ocorrer, houve um foco na responsividade das páginas, para que funcionassem tanto em um navegador de computador quanto no navegador do celular.

Para elaboração da interface visual e fluxo de interação, procurou-se estudar aplicações existentes que implementam de forma adequada o *Material Design*. Notavelmente, foram utilizadas duas fontes de inspiração: Spotify e a página do Firebase, plataforma de *Backend-as-a-Service* fornecida pelo Google. Essa inspiração pode ser notada nas Figuras 11, 12 e 13, que mostram diferentes telas da interface desenvolvida.

É importante ressaltar que, como um exemplo de fluxo de interação, a interface não contempla todas as possibilidades oferecidas pela API, apenas um fluxo básico de uso. A interface apresenta sua própria lógica separada da mesma, utilizando notavelmente duas bibliotecas explicadas a seguir.

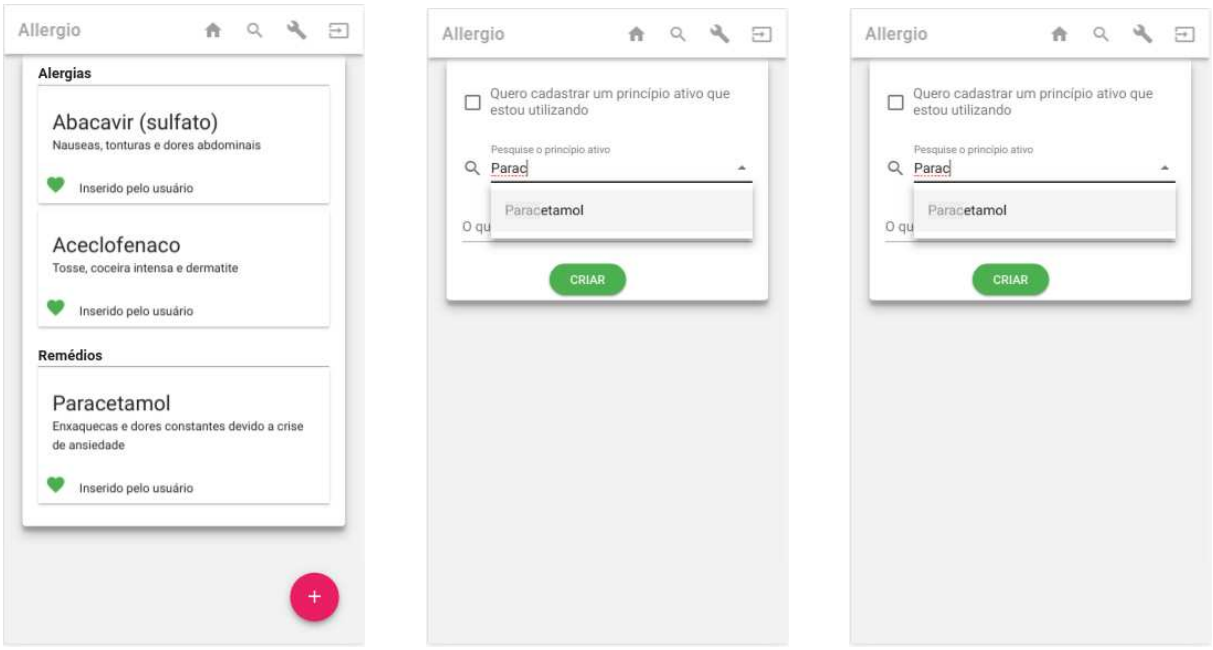


Figura 11 – Tela inicial da interface.

Figura 12 – Tela de cadastro de uma reação adversa.

Figura 13 – Tela de pesquisa para verificar se um medicamento é seguro para consumo.

4.3.1 Vuex

Vuex é uma biblioteca para gerenciamento do estado da aplicação. Fornece um caminho único para manutenção dos dados que serão mantidos pela biblioteca, utilizando os seguintes conceitos (VUEX, 2018):

- *State*: Representa o estado da aplicação em um dado momento. Todos os dados que o usuário deseja guardar com o Vuex estarão aqui e disponível para todos os componentes da aplicação, que podem disparar uma ação;
- *Action*: Uma *action* representa uma ação que tem como objetivo final alterar os dados guardados no *State*. Porém, podem conter lógicas antes de efetivamente alterar os dados. Em seu final, realiza o *commit* de uma *mutation*;
- *Mutation*: Uma *mutation* é a única forma de efetivamente alterar os dados do *State*. Chamada por uma ação que envia os parâmetros a serem guardados no estado, a *mutation* atualiza as informações no mesmo;

Percebe-se o fluxo circular dos dados em uma aplicação que utiliza essa biblioteca. Para eluciar melhor o funcionamento do Vuex, é interessante demonstrar como foi utilizado na interface proposta pelo projeto.

Os dados que são guardados no *State* da aplicação são relativos ao usuário, especialmente o *token* de acesso que deve ser utilizado em toda requisição para a API, praticamente.

Há uma *action* para autenticar um usuário, que faz uma requisição para a API enviando o usuário e senha fornecidos. Caso a autenticação seja bem sucedida, a *action* faz o *commit* da *mutation* de login, que guarda todos os dados enviados pela resposta no estado da aplicação.

De forma análoga, há uma action que realiza o *logout* na interface, fazendo o *commit* de uma *mutation* que apaga todos os dados guardados no estado da aplicação.

Nota-se que na interface, a lógica de estar autenticado ou não é uma mera formalidade de ter as informações do usuário disponíveis para uso, principalmente o *token*.

4.3.2 Vue Router

Vue Router, por sua vez, é uma biblioteca que implementa funcionalidades para facilitar o roteamento interno da aplicação, permitindo a criação de rotas para direcionamento aos componentes desenvolvidos (ROUTER, 2018).

Foi utilizada no projeto com três propósitos:

- Elaboração de rotas para os componentes;
- Redirecionamentos entre os componentes;
- Validação se um usuário está acessando um componente permitido a ele. Se o usuário não estiver autenticado (isso é verificado com os dados fornecidos pelo Vuex), ele só pode acessar a página de cadastro e de *login*. Caso esteja autenticado, pode visualizar as páginas internas da aplicação;

De forma geral, há uma sinergia entre o Vuex, Vue Router e a API. Se o usuário faz um *request* com um *token* expirado, por exemplo, o Vuex automaticamente apaga os dados de usuário guardados no estado da aplicação, efetivamente realizando o *logout* e o Vue Router redireciona o mesmo para a página de *login*.

5 Conclusão

Por fim, o tema a que se refere este projeto é extremamente importante para a saúde das pessoas. Ter como informar as reações adversas de um indivíduo constitui uma forma simples, porém eficiente, de evitar acidentes com essa temática.

Por isso, muitos *softwares* tentam fazer isso de alguma forma, como o Apple Healthkit. Porém, são soluções que podem não ser adotadas de forma oficial, o que pode diminuir sua eficiência.

Ao criar, com o Allergio, um ecossistema em que um usuário pode inserir seus dados e qualquer médico ou enfermeiro pode verificar essas informações, e até incrementá-las, espera-se obter uma solução que incentive informações sempre atualizadas e disponíveis de qualquer lugar, quando forem necessárias.

Adicionalmente, realizar isso como um projeto de *software* livre incentiva que pessoas de áreas diferentes possam conhecer mais sobre o assunto e oferecer novas possibilidades e soluções, engajando ainda mais a discussão sobre o tema.

Por isso, é importante manter a documentação do projeto correta e completa, além de garantir a qualidade do projeto, tanto a nível de funcionalidade quanto a nível de código, tornando o mesmo o mais manutenível possível. A qualidade do código no projeto é garantida através dos testes automatizados e teste estático de código.

Outro aspecto relevante sobre a entrega do *software* diz respeito à escalabilidade do mesmo. A plataforma da AWS utilizada no *deploy* garante a escalabilidade de acordo com triggers configuráveis para que mais instâncias da aplicação possam ser criadas ou destruídas sob demanda (AMAZON, 2018).

5.1 Trabalhos Futuros

Com o DigiSUS, que é a alternativa de *e-Health* para o Brasil (SAúDE, 2018), tem-se uma grande oportunidade de integrar o trabalho desenvolvido neste documento com essa plataforma, de forma a tornar a adoção do *software* oficial e portanto, mais eficiente para prevenção de medicação errônea para pacientes de todo o território brasileiro.

Posteriormente, há diversas possibilidades que podem tornar o projeto cada vez mais utilizado e completo. Como exemplo de funcionalidades que podem incrementar o valor do projeto, pode-se citar:

- Adicionar bulas dos remédios, para que possam ser lidas caso seja necessário;

- Utilizar o projeto como uma forma de abrir dados, de forma eletrônica e consumível, de:
  - Remédios;
  - Princípios ativos;
  - Hospitais brasileiros;

As possibilidades são diversas e o momento propício para integrar cada vez mais novos serviços para melhorar o panorama geral da saúde brasileira.

Referências

ALVAREZ, R. C. The promise of e-health - a canadian perspective. 2002. Citado na página 28.

AMAZON. *O grupo do Auto Scaling de seu ambiente do AWS Elastic Beanstalk*. 2018. <https://docs.aws.amazon.com/pt_br/elasticbeanstalk/latest/dg/using-features.managing.as.html>. Acessado em: 24/03/2019. Citado na página 61.

AMAZON Elastic Beanstalk. 2018. <https://aws.amazon.com/pt/elasticbeanstalk/?nc2=h_m1>. Acessado em: 05/06/2018. Citado na página 46.

ANVISA. *DCB - Definições*. 2018. <http://portal.anvisa.gov.br/dcb/conceitos-e-definicoes>. Acessado em: 20/06/2018. Citado na página 25.

BERND, L. A. Alergia a Medicamentos :. p. 1–8, 2010. Citado na página 25.

BERNERS-LEE, T. et al. Hypertext transfer protocol – http/1.0. 1999. Citado na página 32.

CASSIANI, S. H. D. B. et al. Aspectos gerais e número de etapas do sistema de medicação de quatro hospitais brasileiros. *Revista Latino-Americana de Enfermagem*, v. 12, n. 5, p. 781–789, 2004. ISSN 0104-1169. Citado na página 25.

FIELDING, R. T. Architectural Styles and the Design of Network-based Software Architectures. *Building*, v. 54, p. 162, 2000. ISSN 1098-6596. Disponível em: <https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>. Citado 2 vezes nas páginas 30 e 31.

GITLAB. *Create*. 2018. <https://about.gitlab.com/product/create/>. Acessado em: 10/06/2018. Citado na página 44.

GITLAB. *Getting started with GitLab CI/CD*. 2018. <https://docs.gitlab.com/ee/ci/quick_start/>. Acessado em: 10/06/2018. Citado na página 44.

GITLAB. *GitLab CI/CD Pipeline Configuration Reference*. 2018. <https://docs.gitlab.com/ee/ci/yaml/>. Acessado em: 27/03/2019. Citado na página 49.

GITLAB. *Introducing the GitLab Issue Board*. 2018. <https://about.gitlab.com/features/issueboard/>. Acessado em: 10/06/2018. Citado na página 45.

INTRODUCTION. 2018. <https://vuejs.org/v2/guide/>. Acessado em: 08/06/2018. Citado na página 45.

INTRODUCTION to JSON Web Tokens. 2018. <https://jwt.io/introduction/>. Acessado em: 11/06/2018. Citado na página 46.

KUROSE, J. F.; ROSS, K. W. *Redes de Computadores e a Internet*. [S.l.]: Pearson Education do Brasil, 2010. Citado 3 vezes nas páginas 29, 31 e 32.

MATERIAL Design Component Framework. 2018. <https://vuetifyjs.com/pt-BR/>. Acessado em: 08/06/2018. Citado na página 45.

MELO, D. O. d.; RIBEIRO, E.; STORPIRTIS1, S. A importância e a história dos estudos de utilização de medicamentos. *Revista Brasileira de Ciências Farmacêuticas*, v. 42, n. 2003, p. 475–485, 2006. ISSN 1516-9332. Disponível em: <http://www.revistas.usp.br/rbcf/article/viewFile/44155/47776>. Citado na página 27.

NAGAO-DIAS, A. T. et al. Reações alérgicas a medicamentos. *Jornal de Pediatria*, v. 80, n. 4, p. 259–266, 2004. ISSN 0021-7557. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0021-75572004000500004&lng=pt&nrm>. Citado na página 25.

OH, H. et al. What is ehealth (3): A systematic review of published definitions. 2005. Citado na página 28.

RADIGAN, D. *Kanban*. 2018. <https://www.atlassian.com/agile/kanban>. Acessado em: 28/05/2018. Citado na página 37.

RAILS. *Getting Started with Rails*. 2018. <http://guides.rubyonrails.org/getting_started.html>. Acessado em: 08/06/2018. Citado na página 45.

ROUTER, V. *Introduction*. 2018. <https://router.vuejs.org/>. Acessado em: 10/10/2018. Citado na página 59.

SAúDE, M. da. *Estratégia de Saúde Digital (e-Saúde) para o Brasil: digiSUS*. 2018. <http://portalms.saude.gov.br/acoes-e-programas/digisus>. Acessado em: 24/02/2019. Citado na página 61.

Sociedade Brasileira de Informática em Saúde. Manual de Certificação para Sistemas de Registro Eletrônico em Saúde. *Editor: Marcelo Lúcio da Silva*, p. 91, 2013. Disponível em: <http://www.sbis.org.br/certificacao/Manual_Certificacao_SBIS-CFM_2013_v4-1.pdf>. Citado 6 vezes nas páginas 19, 27, 28, 39, 40 e 45.

VICENTINI, C. F. et al. PEHS – Arquitetura de um Sistema de Informação Pervasivo para Auxílio às Atividades Clínicas. *Revista Brasileira de Computação Aplicada*, v. 2, n. 2, p. 69–80, 2010. ISSN 2176-6649. Disponível em: <http://www.upf.br/seer/index.php/rbca/article/view/970/783>. Citado 2 vezes nas páginas 19 e 28.

VUEX. *What is Vuex?* 2018. <https://vuex.vuejs.org/>. Acessado em: 10/10/2018. Citado na página 58.

WATSON, R. Eu wants every member to develop a “roadmap” for ehealth. 2004. Citado na página 28.

WHAT is a Container. 2018. <https://www.docker.com/what-container>. Acessado em: 05/06/2018. Citado na página 46.

World Health Organization. Safety of medicines. *British Medical Journal*, v. 4, n. 5834, p. 192, 2002. ISSN 00071447. Citado 2 vezes nas páginas 25 e 27.

World Health Organization. mHealth: New horizons for health through mobile technologies. *Observatory*, v. 3, n. June, p. 66–71, 2011. ISSN 2093-3681. Disponível em: <http://www.webcitation.org/63mBxLED9>. Citado na página 28.